This document is published in:

Nguyen , N. T. (2013). *Transactions on Computational Collective Intelligence IX*. (Lecture Notes in Computer Science, 7770), Springer, 79-97.
DOI: http://dx.doi.org/10.1007/978-3-642-36815-8_4

Evaluation of Agents Interactions in a Context-Aware System

Nayat Sanchez-Pi, David Griol, Javier Carbo, and Jose M. Molina

Carlos III University of Madrid,
Avda de la Universidad Carlos III, 22. Colmenarejo. Madrid
{nayat.sanchez,david.griol,javier.carbo,
josemanuel.molina}@uc3m.es

Abstract. The evaluation of Multi-Agent Systems (MAS) is a complex problem and it does not have a single form. Much effort has been spent on suggesting and implementing new architectures of MAS. Often these new architectures are not even compared to any other existing architectures in order to evaluate their relative benefits. The present work focuses on interactions, the most important characteristic of any complex software as autonomous agents according to [25], as a problematic of evaluation. So, in this paper, we describe the assignment of evaluation values to Agents interaction in a specific MAS architecture. This eval-uation is based on the weight of the messages brought by an interaction.

Keywords: Multi-Agent Systems, Evaluation, Context-Aware Architectures.

1 Introduction

The use of nowadays technology like notebooks, PDAs and smart phones gave birth to the concept of mobile computing. Mobile computing field had an increasing attention few years ago, as many systems were designed towards this direction and to be context-aware with the aim of optimizing and automating the distribution of their services in the right time and in the right place. There is another concept introduced first by Weiser [24]: pervasive computing, referring to the seamless integration of devices into the users everyday life. It is also named ubiquitous computing, so devices should vanish into the background to focus on the user and his tasks rather than being aware of the computing devices and technical issues.

Multi-Agent Systems (MAS) appear in Computer Science as one important field in the wide range of pervasive computing. An agent is a computational process that has control over its internal state and behaviour. Software agents provide then an ideal mechanism for implementing heterogeneous and complex distributed systems. A multi-agent system can be defined as a network of software agents that work together and share social skills. Multi-agent systems belong to the field of artificial intelligence, which aims to provide principles for building complex systems involving multiple agents and also the necessary mechanisms to coordinate individual behaviour of agents [23].

Due to advances in communication technologies such as sensor networks and radio frequency identification (RFID), ubiquitous computing is increasingly entering in every aspects of our lifes, opening a world of unprecedented scenarios where users interact

with electronic devices embedded in environments that are sensitive to the presence of users [14]. These context-aware MAS combine ubiquitous information, communication, with enhanced personalization, natural interaction and intelligence. The use of this context offers the possibility to tailor a new type of advanced applications. Context-aware MAS should be able to adapt their operations to the current context without explicit user intervention and taking environmental context into account. Particularly when it comes to using mobile devices, as context data may change so rapidly, it is desirable that programs and services react specifically to their current location, time and other environment attributes and adapt their behaviour according to the changing circumstances.

There are several works addressing evaluation in MAS and are characterized i.e., for its architectural style [5]; for its agent-oriented methodologies based on the software engineering related criteria and characteristics of MAS [15]; [8] or for the complexity of interactions [12]. Due to the distributed nature of agent systems and the complexity of the interaction inside them, their evaluation is a difficult task. According to citewooldridge2001, interactions are the most important characteristic to evaluate in any complex software as autonomous agents.

In this paper, we suggest an assignment of evaluation values to Agents interaction in an specific MAS architecture for providing context services. This evaluation is mainly based on the relevance of the messages content brought by an interaction. For dependant nature of the relevance of the messages, the valuation has to be adhoc, but here we provide an example of how interesting is this alternative in order to evaluate any MAS architecture theoretically.

Other researchers have also focused on interactions-based evaluations. These studies have verified that this kind of evaluation originates different types of problems [12]. Firstly, the effect of an interaction unit (a single message) in an agent system could be equivalent to the definition of n units (messages) in another system. This way, the weight assigned to the same interaction is 1 in the first system and is n in the second. Secondly, the interaction units that are received and cannot be used by an agent could be a bias in the measurement of interaction in MAS.

Our proposal is based on [12]. The first task is to classify the possible received messages into different message types. Then, a weight is associated to a message according to its type. If two messages with the same type produced very different effects on the agent, then this assignment would not provide a correct solution. So we also consider the cost of processing a message and the cost of deciding the corresponding responsive action. They consists in the memorization that needs such message, in terms of change of internal states caused by the received message, and the decision that concerns the choice of the responsive action.

The rest of the paper is structured as follows. Section 2 describes related work on the evaluation of Multi-Agent systems. In section 3 we briefly describe our agent-based architecture that provides context-aware services. Section 4 explains the evaluation method based on the weight of the information brought by a message. Section 5 presents the application of the evaluation method to our MAS. Finally we draw some conclusions and we suggest some future possible directions of research.

2 Related Work: Evaluation of Multi-Agent Systems

Evaluation is a central piece of software engineering. Evaluations methodologies allow researchers to assess the quality of the findings and to identify advantages and disadvantages of a system. The goal in evaluation of conventional systems is proving that a system is more efficient. Normally, variables associated with efficiency are: the time to complete a given task or the number of errors that have been made while fulfilling the task. Evaluating the performance of any system is a complex task but it is even more complex when we deal with distributed systems.

Evaluations for context-aware systems can not be addressed in the same way evaluation is understood for other software systems where the concept of large corpus data, the establishment of ground truth and the metrics of precision and recall are used. Evaluation for changeable systems like context-aware needs to be conducted to assess the impact of the users. The heterogeneity, dynamicity, and heavily context-dependent behaviors of context aware systems require new approaches of performance evaluation. Normally, besides simulation techniques, real-world evaluation is addresed as field studies and relies on collecting data from observation about the usability of the software in the context of use. But there is a need of assessing usability of the software and measure the utility of the context-aware system. A three level approach to evaluation is proposed by [22]: evaluation of the science; evaluation of the component technology, and evaluation of the impact.

Potential system impact metrics include: (a) Trust in the system, (b) Shift in user time, (c) Increased quality of product, (d) Increased confidence in product. Measurement of these attributes will encompass both quantitative and qualitative data. Trust in the system can be measured by the amount of system suggestions that information analysts further explore and by user ratings and comments. Trust in the system may also be a measure of the understanding the user has of how the system actually works. In that respect assessing the user's mental model of the system is appropriate [4,17].

Component metrics are, of course, specific to the particular functionality of the component. To provide some examples of possible metrics, we go back to some current research efforts. In particular, we consider potential metrics for question and answer dialogs, user modeling, hypotheses generation, prior and tacit knowledge. Question and answer systems currently being developed in the research community go beyond simple Question and Answer systems. These systems might help a user construct a travel itinerary, with flight reservations, hotel bookings and car rentals. Evaluation efforts for dialog systems have used user satisfaction as the impact metric. Component metrics for question and answer dialogs might include: (i) Completeness of answer; (ii) Accuracy of answer; (iii) Effort required on part of user engaging in dialog.

Finally, scientific metrics are still important and need to be the first evaluation applied. Consider the same topics discussed in component metrics: question and answer dialogs, user modeling, hypothesis generation, prior and tacit knowledge.

Since some years, there is a growing interest in understanding specific evaluation problems that arise from context-aware systems [21,6,22]. For instance, [20] proposes a taxonomy where we can find several methods for evaluating context aware systems based on Pre-implementation Evaluation, Sub-system Evaluation, and Overall System Evaluation. Research groups such as the Future Computing Environments Group at

Georgia Institute of Technology working on the "Aware Home" [1] and Tatsuya Yamazaki of National Institute of Information and Communications Technology, Japan working on the "Ubiquitous Home" [26] have completed real-life test home environments for accurate simulation of the home environment. Both groups aim to perfectly emulate a real domestic environment and intend to have test-subjects spend significant periods of time in these simulated home environments carrying out domestic activities. However, such live usage test beds are expensive and difficult to reconfigure to emulate a wide range of different contexts.

Kerttula and Tokkonen [13] have identified "the total user experience" as an area of concern and aim to achieve it through early product and system simulations. This idea moves away from testing in isolation and moves towards a simulation where services are tested in parallel and valued over longer periods of time. This approach uses accurate simulation/prototyping of services focusing on features such as the user interface, audio properties and product behaviour, but not including the user's surrounding physical environment.

Huebscher and McCann [10] aim to allow initial testing of context-aware applications without requiring a physical deployment. However Huebscher and McCann are working to simulate sensor data e.g. temperature, humidity or location, from a description of context or a simulation model of contexts. This in turn will be used to test the context-logic of a context-aware application.

In the past, virtual reality simulation of pervasive computing environments has been used in a small number of research efforts, specifically QuakeSim [3] and HP Lab's UbiWise [2]. These have demonstrated that 3D virtual reality computer game engines potentially provide a cost effective platform for simulating pervasive computing environments with sufficient realism to accurately test human interaction with pervasive computing software systems.

Also, Shirehjini and Klar have been developing 3DSim [16], a 3D tool for rapidly prototyping Ambient Intelligence building blocks e.g. situation-recognition, goal-based interaction. 3DSim aids the development of human-ambient-interaction systems such as PDA based control systems, adaptive user interfaces, multimedia output coordination or goal-based interaction systems. During a simulation, sensor data is derived from a 2D GUI and gesture elements which are the result of an avatar pointing at devices.

The team at GIST U-VR Lab, S. Korea have been working on creating a unified context model and a method for the integration of contexts for unified context-aware applications. To loosen the coupling between services and context, they have developed a unified context that represents user-centric contextual information in terms of 5W1H (Who, What, Where, When, How and Why) [11]. To demonstrate user-centric integration of contexts for a unified context-aware application model (the ubi-UCAM), they created a simple 3D simulated environment [18]. By using the simulator they were able to test the effectiveness of the Context Integrator when there were multiple users working with the service simultaneously. The simulated environment allowed them to assess the capabilities of their Context Integrator before bringing it into a real world situation (ubiHome).

Other examples of context-aware system evaluations are in Ambient Intelligence (AmI) where sensory information is used to determine the context of a situation and

the application modifies its behaviour accordingly. The behaviour could be an action performed automatically or information that is modified based on the context of the user. Other researchers in augmented cognition and affective computing are attempting to understand the cognitive state of the user and modify the delivery of information based on what the user can currently handle. Evaluations of such systems cannot be done solely through empirical methods as there are too many situations to test. Simulation could be used as a supplement though human judgment of the appropriateness of the modified behaviour needs to be the ultimate metric [3].

In this paper, we focus on the consideration of interaction as the main parameter of evaluation, based on several research studies like the one [12]. As stated in the introduction, several tasks are required to perform this evaluated based on the classification of the possible messages in the system into specific sets sharing the same type, the assignment of a weight to a message according to its type, and the evaluation of the change at the internal state caused by the received messages and the decisions that concern the choice of the corresponding actions.

3 Our Agent-Based Context-Aware System

The proposed agent-based architecture manages context information to provide personalized services through users interactions with conversational agents. As it can be observed in Figure 1, it consists in five different types of agents that cooperate to provide an adaptive service. *User agents* are configured into mobile devices or PDAs. Providers are implemented as *Conversational Agents* that provide the specific services through dialogs with users. A *Facilitator Agent* matches the different positions and interests of users with the providers defined in the system. A *Positioning Agent* communicates with the ARUBA positioning system [19] to obtain and provide positioning information of user agents in the system. Finally, a *Log Analyzer Agent* generates user profiles from the users' previous dialogs that are used by Conversational Agents to adapt their behaviour.

A conversational agent includes a software that accepts natural language as input and generates natural language as output, engaging in a conversation with the user. To successfully manage the interaction with the users, conversational agents usually carry out five main tasks: automatic speech recognition (ASR), natural language understanding (NLU), dialog management (DM), natural language generation (NLG) and text-to-speech synthesis (TTS). These tasks are usually implemented in different modules. In our architecture, we incorporate a Context Manager in the architecture of the designed conversational agents, This module deals with loading the context information provided by the User and Positioning Agents, and communicates it to the different modules of the Conversational Agent during the interaction.

To manage context information we have defined a data structure called *user profile*. Context information in our user profile can be classified into three different groups. *General user information* stores user's name and machine identifier, gender, preferred language, pathologies or speech disorders, age, *Users Skill level* is estimated by taking into account variables like the number of previous sessions, dialogs and dialog turns, their durations, time that was necessary to access a specific web service, the date of the last interaction with the system, etc. Using these measures a low, medium, normal, high

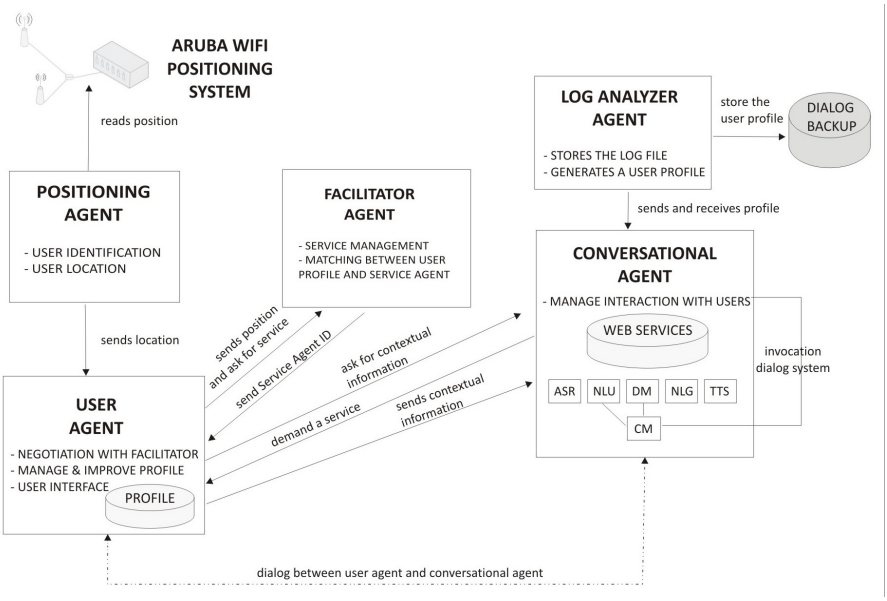

Fig. 1. The proposed multi-agent architecture

or expert level is assigned. *Usage statistics and preferences* are automatically evaluated taking into account the set of services most required by the user during the previous dialogs, date and hour of the previous interactions and preferred output modality.

The free software JADE (Java Agent Development Framework)[1] has been used for the implementation of our architecture. It was the most convenient option as it simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications and through a set of graphical tools that supports the debugging and deployment phases. The agent platform can be distributed across machines and the configuration can be controlled via a remote GUI.

3.1 Ontology

Jointly to our system architecture, we defined an objective description of the concepts and relationships of the domain of knowledge used in the exchanged messages by agents. This explicit and formal specification of a shared conceptualization is what we usually called ontology [9]. An ontology allows that a content of a message can be interpreted unambiguously and independently from the context.

Eight concepts have been defined for the generic ontology used in our system. The definition is: *Location* (*XCoordinate* int, *YCoordinate* int), *Place* (*Building* int, *Floor* int), *Service* (*Name* String), *Product* (*Name* String, *Characteristics* List of Features),

[1] `http://jade.tilab.com/`

Feature (*Name* String, *Value* String), *Context* (*Name* String, *Characteristics* List of Features), *Profile* (*Name* String, *Characteristics* List of Features), *DialogLog* (*Log* List of Features).

Our ontology also includes six predicates with the following arguments: *HasLocation* (*Place*, *Position*, and *AgentID*), *HasServices* (*Place*, *Position*, and *List of Services*), *isProvider* (Place, Position, AgentID, Service), *HasContext* (*What*, *Who*), *HasDialog* (*DialogLog* and *AgentID*), *HasProfile* (*Profile* and *AgentID*), and *Provide* (*Product* and *AgentID*).

3.2 Agents Interaction

The interaction with the different agents follows a process which consists of the following phases:

1. The ARUBA positioning system is used to extract information about the positions of the different agents in the system. This way, it is possible to know the positions of the different User Agents and thus extract information about the Conversational Agents that are available in the current location.
2. The Positioning Agent reads the information about position (coordinates x and y) and place (*Building* and *Floor*) provided by the ARUBA Positioning Agent by reading it from a file, or by processing manually introduced data.
3. The Positioning Agent communicates the position and place information to the User Agent.
4. Once a User Agent is aware of its own location, it communicates this information to the Facilitator Agent in order to find out the different services available in that location.
5. The Facilitator Agent informs the User Agent about the services available in this position .
6. The User Agent decides the services in which it is interested.
7. Once the User Agent has selected a specific service, it communicates its decision to the Facilitator Agent and queries it about the service providers that are available.
8. The Facilitator Agent informs the User Agent about the identifier of the Conversational Agent that supplies the required service in the current location.
9. The User Agent asks the Conversational Agent for the required service.
10. Given that the different services are provided by context-aware Conversational Agents, they ask the User Agent about the context information that would be useful for the dialog. The User Agent is never forced to transmit its personal information and preferences. This is only a suggestion to customize the service provided by means of the Conversational Agent.
11. The User Agent provides the context information that has been required.
12. The conversational agent manages the dialog providing an adapted service by means of the context information that it has received.
13. Once the interaction with the Conversational Agent has finished, the Conversational Agent reads the contents of the log file for the dialog and send this information to the Log Analyzer Agent.
14. The Log Analyzer Agent stores this log file and generates a user profile to personalize future services. This profile is sent to the Conversational Agent.

4 Evaluation Proposal Based on Agents Interaction

The consideration of interaction as a problematic of evaluation has been addressed for other researches. But such an evaluation of interaction brings up different types of problems [12]. First, the effect of an interaction unit (a single message) in an agent system may be equivalent to n units (messages) in another system. The weight of the interactions realizing the same work is 1 and it is n in the second. Later, the interaction units that are received and cannot be used by an agent may be a bias in the measurement of interaction in MAS

Therefore, we take the idea of [12]. The first idea is to classify the possible received messages into sets having the same type. Then, a weight is associated to a message according to its type. But when two messages of the same type have very different effects on the agent, this idea does not provide with a correct solution. The effects of interactions consist of the processing of the message and then a responsive action. The processing is realized by: memorization that treats the part of change at the internal state caused by the received message, and the decision that concern the choice of the action that will be handled. In this work, we follow this approach where the evaluation is based on the weight of the information brought by a message is suggested.

According to the model, two kinds of functions are considered:

- A function $Interaction$ associates weight to the message according to its type. Function $Interaction$ can be computed adopting the primitives proposed by [7] to the type of interaction. This work consists of four possibilities of message types: present, request, answer, and inform. We note $M^A_{received}$ as the set of messages which may be received by agent A, and the function $Interaction$ associates a weight value for each received message by agent A:

$$Interaction = M^A_{received} \rightarrow weightofmessagetype \quad (1)$$

- This solution partially resolves the problem and it works when two messages of the same type have equivalent effects on the agent. So, we introduce Φ that associates weight to the message according to the change provoked on the internal state and the actions triggered by its reception. This function evaluates the treatment of a message in agent systems. For better understanding Φ is divided into two functions: one that evaluates Decision, DD^A and another that evaluates Memorization, MM^A. The function MM^A associates a value to the variation of the internal state (caused by message received). We note as So^A the set of possible original internal states for the agent A while Sf^A is the set of final internal states for agent A. To quantify, some measurable characteristics of the internal state must be defined. The specification of these characteristics is related to the application domain. Since these characteristics have an associated weight, then the function MM^A is considered as the sum of these weights:

$$MM^A = So^A \times Sf^A \rightarrow sumofweightsofstatecharacteristicsthatchanged \quad (2)$$

  Concerning DD^A, this function associates a value to the triggered actions (results of the message received). To quantify, certain type of actions must be defined.

A type of actions having a weight. Then, the value of the function DD^A is considered as the sum of the weights of triggered actions where A^A stands for the set of actions may be done by agent A:

$$DD^A = A^A \rightarrow sumofweightsoftriggeredactions \tag{3}$$

Then, the function Φ dependant of the change of internal state and of triggered actions due to the message received, is then defined as the sum of these functions DD^A and MM^A.

The approach then focuses on the evaluation of interactions in MAS based on this function combination: $Interaction + \Phi$.

5 Evaluating Our MAS

In this section we present the application of the interaction-based evaluation method described above to the architecture presented in Figure 1. It was completely implemented using JADE/LEAP Agents and ARUBA positioning system (see screen capture of Figure 2). In order to apply evaluation, we have computed functions and assigned weights to each message for each agent interaction.

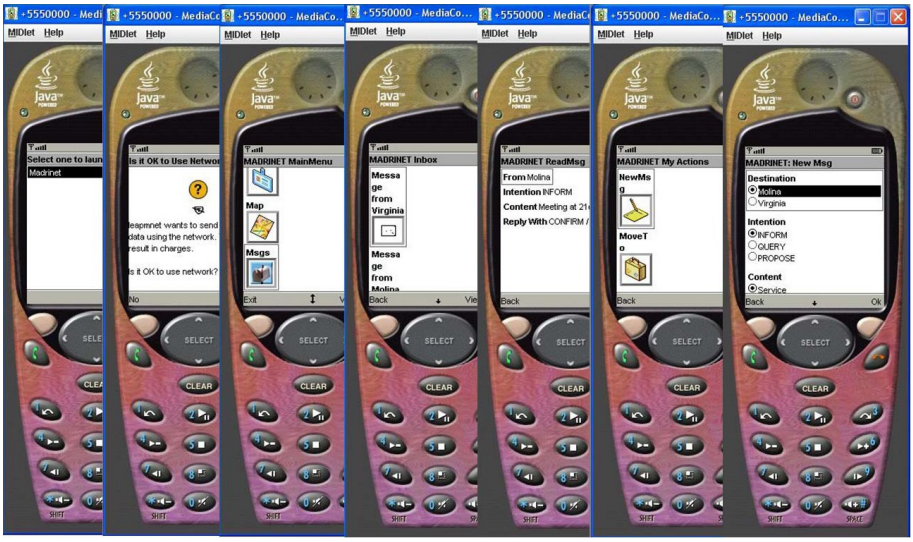

Fig. 2. Screen Capture of JADE/LEAP agent of our system running in an airport domain

5.1 Weights vs. Type of Message: Function *Interaction*

First we compute function *Interaction* according to the four possibilities of message types mentioned in Section 3: present, request, answer, and inform. Each message of phases is classified into one of them:

1. The ARUBA positioning system knows a change in position of a given User agent: *no message involved*
2. The Positioning Agent reads from a file the information about position provided by the ARUBA Positioning Agent: *no message involved*
3. The Positioning Agent communicates the position (coordinates x and y) and place (Building and Floor) information to the User Agent: *present message*.
4. Once a User Agent is aware of its own location, it asks about the different services available in that location to the Facilitator Agent: *request message*.
5. The Facilitator Agent informs the User Agent about the services available in this position: *inform message*
6. The User Agent decides the services in which it is interested: *no message involved*
7. Once the User Agent has selected a specific service, it communicates its decision to the Facilitator Agent and queries it about the service providers that are available: *request message*
8. The Facilitator Agent informs the User Agent about the identifier of the Conversational Agent that supplies the required service in the current location: *inform message*
9. The User Agent asks the Conversational Agent for the required service: *request message*
10. Conversational Agent ask the User Agent about the context information that would be useful for the dialog: *request message*
11. The User Agent provides the context information that has been required: *inform message*
12. The conversational agent manages the dialog providing an adapted service by means of the context information that it has received: *no message involved*
13. Once the interaction with the Conversational Agent has finished, the Conversational Agent send the log file of the dialog to the Log Analyzer Agent: *inform message*
14. The Log Analyzer Agent stores this log file and generates a user profile to personalize future services. This profile is sent to the Conversational Agent: *inform message*

These four types have to be distinguished because of the different basic behaviors that they model from the sender or the receiver points of view:

- A request includes a change of state of the sender, waiting for the answer.
- An inform includes no change of state for both the sender and the receiver. It might generate other informs, and possibly answers.
- A present includes a possible change in the state of the sender and/or of the receiver. Typically, a present will enable entering a society and introduce itself to other agents

5.2 Weights vs. Treatment of a Message: Function Φ

Next we have to evaluate the function Φ that computes the variation of internal state caused by memorization step and decision step. Memorization is evaluated by function

MM^A, since we have an ontology described in Section 3 we can measure that changes according to:

- number of concepts involved
- number of attributes involved

We could also consider the relevance of attributes and concepts giving different weights to any of them. We consider three levels of relevance: low, medium and high. For instance:

- Phase 1. ARUBA localizes a given user: no attributes involved since no message exchange involved (0 required attributes).
- Phase 2. Positioning agent receives internally location information from ARUBA: no attributes involved since no message exchange involved (0 required attributes).
- Phase 3. Positioning agent sends location information to User Agent: attributes relative to location involved (4 required attributes: Building, Floor and Coordinates X Y, relevance: *medium*)
- Phase 4. User Agent asks about available services to a Facilitator Agent: attributes relative to location involved (4 required attributes: Building, Floor and Coordinates X Y, relevance: *medium*)
- Phase 5. Facilitator Agent informs User Agent about Services available, it is where the Facilitator Agent informs the User Agent about the services available at this position: It depends on the number of services (s). Particularly: $2s$ attributes required (s times a service type, s times a service name, relevance: *medium*)
- Phase 6. User agent decides internally which service is interesting. No attributes involved since no message exchange involved (0 required attributes).
- Phase 7. User agent informs the chosen service to the Facilitator Agent: 2 required attributes Service type and Service name, relevance: *medium*
- Phase 8. Facilitator agent informs the user agent about the Conversational Agents who provide that service: it depends on the number of conversational agents who provide similar services (diversity of services), we call this factor (*level of overlapping of services*) that requires 1 attribute per Facilitator Agent (Agent global name)
- Phase 9. User Agent asks Conversational Agent about the required service:2 required attributes Service type and Service name, relevance: *medium*
- Phase 10: Conversational Agent requests context information about the service requested to the User Agent. Since the utility of the context-aware system depends directly on the personal information and preferences that the User Agent freely decides to communicate, the relevance is *high*. This is only a suggestion to customize the service provided by means of the Conversational Agent. Weights linked to this exchanged context information depend on the the attributes required by the chosen service (na_s).
- Phase 11: User Agent provides the context information that has been requested by Conversational Agent: Similarly to Phase 10, Weights linked to this exchanged context information depend on the the attributes required by the chosen service (na_s), relevance is again *high*.

- Phase 12: Conversational Agent provides the service using received context information. no attributes involved since no message exchange involved (0 required attributes).
- Phase 13: Conversational Agent sends the log file to the Log Analyzer Agent: Log file contains the different attributes of the complete context information corresponding to the provided service. Their number depends on the the attributes required by the chosen service (na_s), and since adaptivity of the context-aware system relies on the availability of several logs per user, the relevance of the corresponding attributes is *high*.
- Phase 14: Finally a *high* weight is assigned to the exchanged messages of this phase where the Log Analyzer Agent sends a user profile to the Conversational Agent. Since this profile is generated by the aggregation of several dialog logs to personalize future services, it can be assumed a high relevance to this message exchange. Number of attributes involved depends on the overlapping of this log respect to the previous dialogs implemented (partly time-decrescent function). We call this factor *level of log overlapping*

Then we need to compute DD^A function that associates the variation of internal state caused by decision step. This function associates a value to the triggered actions. In order to quantify it, certain type of actions must be defined. Each type of actions should have a weight. Then, the value of the function DD^A is considered as the sum of the weights of triggered actions. Again we classify the weight in three categories: low, medium and high. The set of actions involved in our agent system can be also classified as external and internal. The external actions correspond to the communicative responses to the given message, where the weight of this reactive action is obtained from the weight of the content included in the responsive messages. On the other hand, internal actions involve the processing and decision making of the next phases:

- Phase 4: Once a User Agent is aware of its own location, it asks about the different services available in that location to the Facilitator Agent: request message. Simple query to the internal database of Facilitator agent. No intelligence involved: *low weight*.
- Phase 6: The User Agent decides the services in which it is interested. Intelligent and relevant decision with a real economic cost: *high* weight.
- Phase 8: The Facilitator Agent informs the User Agent about the identifier of the Conversational Agent that supplies the required service in the current location: Simple query to the internal database of Facilitator agent. No intelligence involved: *low* weight.
- Phase 11: The User Agent provides the context information that has been required. Intelligent and relevant decision with a privacy cost: *high* weight.
- Phase 13: Once the interaction with the Conversational Agent has finished, the Conversational Agent send the log file of the dialog to the Log Analyzer Agent. Simple query to the internal database of Conversational agent. No intelligence involved: *low* weight.
- Phase 14: The Log Analyzer Agent stores this log file and generates a user profile from it to personalize future services. Intelligent and relevant decision: *medium* weight.

5.3 Case of Use: a Montecarlo Simulation of an Airport Context-Aware System

In order to define a case of use, we used a scenario inspired in the Airport domain which uses an extension of the generic ontology mentioned in Section 2 [19].

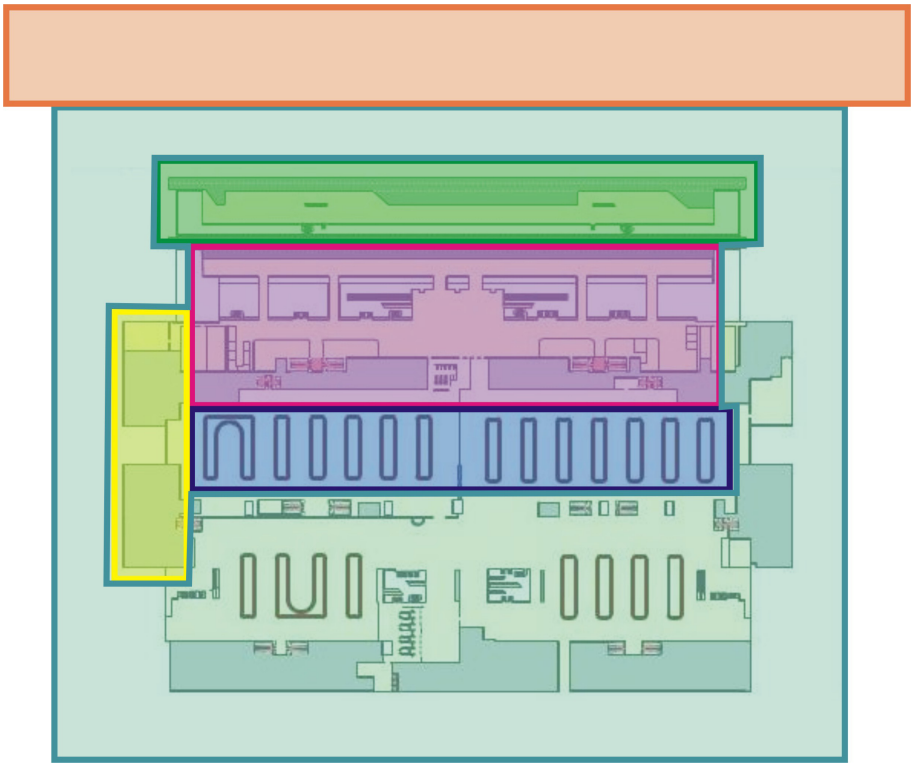

Fig. 3. Zones of the Airport

Our airport scenario has six different zones that do not overlap with each other. Figure 3 shows the map of the airport with the zones that host services of our context-aware system. They are: Airport Zone (register service), Customs Zone (customs service), Commercial Zone (magazines, restaurants, spa, shop), Offices Zone (airline offices services), Check-In Desk Zone (check-in service) and Finger Zone (finger service). Providing each of these services involves filling a given number of concept attributes belonging to the airport Ontology. The number of involved attributes is highly variable (from the simplest services such as register and finger to the most complex ones in the commercial zone). They are:

- register: flight number (required), passenger name (optional), passenger nationality (optional) checkin zone (required)
- customs: anything to declare (required), value of declared objects (optional), nationality (optional)

– checkin: size of baggage (required), weight of baggage (optional), flight number (required), window/aisle (optional), preferred row (optional), food special (optional), finger number (required)
– offices: flight plan (required), accreditation (required), weather conditions (optional)
– finger: special needs (optional)
– magazines: type (optional), language (optional), price (required), name (required)
– restaurants: food (optional) drink (required) price (required)
– spa: type (required) length (optional) price (required)
– shop: product (required) price (required)

Table 1. Initial setup of services in the Airport scenario

Service	Service #	Room	Room #	Max Attributes x Service	Min Attr. x Serv.
Register	1	Airport	1	4	2
Customs	2	Customs	2	3	1
Magazines	3	Commercial	3	4	2
Restaurant	4	Commercial	3	2	
Spa	5	Commercial	3	3	2
Shop	6	Commercial	3	2	2
Office	7	Office	4	3	2
Checkin	8	Checkin	5	7	3
Finger	9	Finger	6	1	0

Table 1 shows the initial setup of these services: room where they are located, and range of attributes per service.

Table 2. Possible paths followed by users

Role		Step1	Step2	Step3	Step 4
Outgoing	Services	1	8	3,4,5,6	9
	Rooms	1	5	3	6
Ingoing	Services	1	2	3,4,5,6	
	Rooms	1	2	3	
Pilot	Services	1	8	7	3,4,5,6
	Rooms	1	5	4	3

Also services potentially required by users depend on their roles: Pilots, outgoing passengers, ingoing passengers. Therefore they will follow different paths through sequential steps where user may require one or more services. These paths are shown in Table 2. Each service of these paths involves a cycle of the 14 execution phases of our context aware system as we defined it previously. As we also mentioned in Section 4.1, several types of messages were distinguished. Now we proceed to associate adhoc numerical values to these message types:

– request: 2 (a change of state, a reaction produced)
– inform: 1 (no change of state)
– present: 1.5 (1 or 2 change of state)

Additionally we have to give specific values to the labels: maximum, medium and minimum. We consider weights of 1, 1.5 and 2 respectively. And we can then conclude the final weight of each phase as it is shown in Table 3, where na_s stands for the number of attributes involved in the corresponding agent interaction since it is not a constant value, it is different for each service s (see Table 1).

Table 3. Final weights of agents interactions

Phase	Message type	Internal processing	Attributes Relevance	Attributes involved	Final weight
1	0	0	0	0	0
2	0	0	0	0	0
3	1.5	0	1.5	4	1.5 + 1.5*4
4	2	1	1.5	4	2+1+1.5*4
5	1	0	1.5	2*s	1+1.5*2*s
6	0	2	0	0	2
7	2	0	1.5	2	2+1.5*2
8	1	1	1.5	Service_Overlapping	1+1+1.5*Service_Overlapping
9	2	0	1.5	2	2+1.5*2
10	2	0	2	na_s	2+2*na_s
11	1	2	2	na_s	1+2+2*na_s
12	0	0	0	0	0
13	1	1	2	na_s	1+1+2*na_s
14	1	1.5	2	%overlappingLog	1+1.5+2*%overlappingLog*na_s

If we consider that our Airport domain has no overlapping services, and that the number of services per room is one but in commercial zone, where it is 4, then we will have the total weights for every step involved in these 14 phases. In Table 4 we can observe the associated weights according to the total number of concept attributes included in the agent interactions corresponding to the 14 phases (m= 3*s + 1.5*Service_Overlapping + 2*%overlappingLog*na_s + 6*na_s)), in function of the interest of the particular user in each possible commercial service (s). In our case if we asume that Service_Overlapping=0 and %overlappingLog=1, then m will be 3*s+8*na_s.

Table 4. Weights associated to steps followed by users

Role		Step1	Step2	Step3	Step 4
Outgoing	Services	1	8	3,4,5,6	9
	Rooms	1	5	3	6
	Weights	29+m	29+m	i(29+m)	29+m
Ingoing	Services	1	2	3,4,5,6	
	Rooms	1	2	3	
	Weights	29+m	29+m	i(29+m)	
Pilot	Services	1	8	7	3,4,5,6
	Rooms	1	5	4	3
	Weights	29+m	29+m	29+m	i(29+m)

In order to simulate the interactions involved in this scenario we use montecarlo method to draw some conclusions. Therefore we consider as random variables: the role of a user (a 5% of pilots, and a 47.5% of outgoing and ingoing passagers), the interesting commercial services (for a user) and the final number of attributes involved for each service (which can be less than the maximum and more than the minimum since we have required attributes and optional attributes). Table 5) shows the chosen random distributions for these variables.

Table 5. Random variables in the Airport Scenario

Variable	Probability Distribution	From	To	Values
Role	Uniform	0	1	Pilot (> 0.95), Outgoing (< 0.475), Ingoing (other cases)
Interest in Magazines	Uniform	0	1	False, True
Interest in Restaurants	Uniform	0	1	False, True
Interest in Spa	Uniform	0	1	False, True
Interest in Shop	Uniform	0	1	False, True
# Attributes Register	Uniform	2	4	
# Attributes Customs	Uniform	1	3	
# Attributes Magazines	Uniform	2	4	
# Attributes Restaurants	Uniform	2	3	
# Attributes Spa	Uniform	2	3	
# Attributes Shop	Fixed	2	2	
# Attributes Office	Uniform	2	3	
# Attributes Checkin	Uniform	3	7	
# Attributes Finger	Uniform	0	1	

From the results showed in tables 6, 7, 8, and 9 we can observe the average, standard deviation and the confidence interval at 95% of agent interactions by outgoing passengers, ingoing passengers, pilots and total users respectively.

Table 6. Evaluation results of Agent interactions for 1000 Outgoing passengers

Variable	Value
Average	289,75
Standard Deviation	64,10
Confidence Interval at 95%	(285,78-293,72)
Min	144
Max	444
Estimation error	0,0316

Table 7. Evaluation results of Agent Interactions for 1000 Ingoing passengers

Variable	Value
Average	229,38
Standard Deviation	63,68
Confidence Interval at 95%	(225,44-233,33)
Min	88
Max	372
Estimation error	0,0316

Table 8. Evaluation results of Agent Interactions for 100 Pilots

Variable	Value
Average	294,58
Standard Deviation	59,41
Confidence Interval at 95%	(282,93-306,22)
Min	176
Max	452
Estimation error	0,1

Table 9. Evaluation results of Agent interactions for 2100 Users

Variable	Value
Average	261,24
Standard Deviation	70,54
Confidence Interval at 95%	(258,22-264,25)
Min	88
Max	452
Estimation error	0,0218

Although these results just represent an estimation, we can conclude some statements about the relative weights of interactions from different kind of agents have inside this Context-Aware System. Particularly in our simulation we can observe how pilots produce slightly more overhead of Agent interactions than Outgoing Passengers, while Ingoing Passengers are clearly the most efficient agents in terms of Agent interactions. This information could be for instance used to improve scalability, estimating the maximum number of allowed agents in a system by each type. An alternative use could be redesigning the ontology and protocols of the system to avoid possible future bottlenecks.

Additionally, we could compare in a fair way other agent architectures in the same domain and scenario (for instance a more centralized system, or a system without conversational agents) using these interaction-based evaluation. This is the main intention of this research contribution. We have shown with this particular simulation how the evaluation method proposed can be applied. Although ad hoc valuations were needed, and although the chosen domain is not very complex (number of services per room, existence of overlapping services, etc.), we think it can be universally applied to any domain.

6 Conclusions

Evaluations of MAS can not be addressed in the same way evaluation is understood for other software systems where the concept of large corpus data, the establishment of ground truth and the metrics of precision and recall are used. Evaluation for autonomous systems like agents needs to be conducted to compare architecture alternatives. Normally, besides the simulation techniques, real-world evaluation is conducted

as field studies and relies on collecting data from observation about the usability of the software in the context of use. But there is a need of assessing performance of the software agents and measure the interaction between them.

Evaluating the interactions in Agent Systems allows to compare alternative agent architectures and protocols between them. In this paper, we have described a proposal of evaluation of Multi-Agent Systems (MAS) based on their agents interactions. This approach based on the weight of the exchanged messages by the agents opens a new way to compare and evaluate the efficiency of different systems, architectures, or agents. We have also applied it to an Agent-based Context-Aware System, assigning evaluation values according to it to show how a final valuation can be obtained. Finally, our approach to the Airport domain allows us to experience the problems of giving values to such weights and of defining testing scenarios to their last consequences. Addressing both problems let us justify valuation criterion for weight assignments that are obviously ad hoc. As future work we would aim to include more experimentation on different domains with this evaluation technique.

Acknowledgments. Funded by projects CICYT TIN2008-06742-C02-02/TSI, CICYT TEC2008-06732-C02-02/TEC, SINPROB, CAM MADRINET S-0505/TIC/0255, and DPS2008-07029-C02-02.

References

1. Abowd, G., Atkeson, C., Bobick, A., Essa, I., MacIntyre, B., Mynatt, E., Starner, T.: Living laboratories: the future computing environments group at the Georgia Institute of Technology. In: Proc. of CHI 2000 Extended Abstracts on Human Factors in Computing Systems, pp. 215–216 (2000)
2. Barton, J., Vijayaraghavan, V.: UBIWISE, A Ubiquitous Wireless Infrastructure Simulation Environment. HP LABS (2002)
3. Bylund, M., Espinoza, F.: Testing and demonstrating context-aware services with quake iii arena. Communications of the ACM 45(1) (2002)
4. Carroll, J., Olson, J., Anderson, N.: Mental models in human-computer interaction: Research issues about what the user of software knows. National Academies (1987)
5. Davidsson, P., Johansson, S., Svahnberg, M.: Characterization and Evaluation of Multi-agent System Architectural Styles. In: Garcia, A., Choren, R., Lucena, C., Giorgini, P., Holvoet, T., Romanovsky, A. (eds.) SELMAS 2005. LNCS, vol. 3914, pp. 179–188. Springer, Heidelberg (2006)
6. Dey, A., Mankof, J., Abowd, G., Carter, S.: Distributed mediation of ambiguous context in aware environments. In: Proc. of the 15th Annual ACM Symposium on User Interface Software and Technology, pp. 121–130 (2002)
7. Gaspar, G.: Communication and belief changes in a society of agents: Towards a formal model of autonomous agent. In: Descentralized A. I. 2, pp. 245–255. Elsevier Science, Amsterdam (1991)
8. Giunchiglia, F., Mylopoulos, J., Perini, A.: The tropos software development methodology: Processes, models and diagrams. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 63–74. ACM Press (2002)
9. Gruber, T.: The role of common ontology in achieving sharable, reusable knowledge bases. In: 2nd International Conference on Principles of Knowledge Representation and Reasoning, Cambridge, USA, pp. 601–602 (April 1991)

10. Huebscher, M., McCann, J.: Simulation model for self-adaptive applications in pervasive computing. In: Proc. of 15th International Workshop on Database and Expert Systems Applications, pp. 694–698 (2004)
11. Jang, S., Ko, E., Woo, W.: Unified user-centric context: Who, where, when, what, how and why. Personalized Context Modeling and Management for UbiComp Applications 149 (2005)
12. Joumaa, H., Demazeau, Y., Vincent, J.: Evaluation of multi-agent systems: The case of interaction. In: 3rd International Conference on Information and Communication Technologies: From Theory to Applications, pp. 1–6. IEEE (2008)
13. Kerttula, M., Tokkonen, T.: The total user experience-how to make it positive in future wireless systems and services. In: Proc. of WWRF Annual Workshop (2001)
14. Lyytinen, K., Yoo, Y., Varshney, U., Ackerman, M., Davis, G., Avital, M., Robey, D., Sawyer, S., Sorensen, C.: Surfing the next wave: design and implementation challenges of ubiquitous computing environments. Communications of the Association for Information Systems, 697–716 (2004)
15. Mylopoulos, J., Kolp, M., Giorgini, P.: Agent-oriented software development. In: Vlahavas, I.P., Spyropoulos, C.D. (eds.) SETN 2002. LNCS (LNAI), vol. 2308, pp. 3–17. Springer, Heidelberg (2002)
16. Nazari Shirehjini, A.A., Klar, F.: 3DSim: rapid prototyping ambient intelligence. In: Proc. of the 2005 Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-Aware Services: Usages and Technologies, pp. 303–307 (2005)
17. Norman, D.: The psychology of everyday things. Basic Books (1988)
18. Oh, Y., Lee, S., Woo, W.: User-centric integration of contexts for a unified context-aware application model. In: Proc. of Joint sOc-EUSAI Conference (2005)
19. Sánchez-Pi, N., Fuentes, V., Carbó, J., Molina, J.M.: Knowledge-based system to define context in commercial applications. In: Proc. of the 8th ACIS Conference SNPD 2007, Tsingtao, China, pp. 694–699 (2007)
20. Schmidt, A.: Ubiquitous Computing- Computing in Context. Ph.D. thesis (November 2002)
21. Scholtz, J.: Ubiquitous computing goes mobile. ACM SIGMOBILE Mobile Computing and Communications Review 5(3), 32–38 (2001)
22. Scholtz, J., Consolvo, S.: Toward a framework for evaluating ubiquitous computing applications. IEEE Pervasive Computing 3(2) (2004)
23. Stone, P., Veloso, M.: Multiagent systems: A survey from a machine learning perspective. Autonomous Robots 8(3), 345–383 (2000)
24. Weiser, M.: The computer of the 21st century. Scientific American 265(3), 66–75 (1991)
25. Wooldridge, M.J., Ciancarini, P.: Agent-Oriented Software Engineering: The State of the Art. In: Ciancarini, P., Wooldridge, M.J. (eds.) AOSE 2000. LNCS, vol. 1957, pp. 1–28. Springer, Heidelberg (2001)
26. Yamazaki, T.: Ubiquitous home: real-life testbed for home context-aware service. In: Proc. of First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, Tridentcom 2005, pp. 54–59 (2005)